AMENDMENTS TO THE SPECIFICATION

Please amend paragraphs [0045], [0047], [0054], [0059], [0060], [0061], [0063], [0066], [0116] and [0121] as follows:

[0045] In the discussion of FIG. 2, the PDSN 109a shown in FIG. 2 corresponds to the PSDN 109 shown in FIG. 1. In the discussion of FIGs. 3-8, regarding the principles of the present invention, the PDSN [[109]] 210 shown in FIG. 5 corresponds to the PSDN 109 shown in FIG. 1. The PDSN 109a of FIG. 2 and the PDSN [[109]] 210 of FIG. 5 have some features in common, but those two PDSNs are not identical.

[0047] In the present invention, PPP framing data are subjected to GRE tunneling and exchanged between the MS and the PDSN, and, from among the functions of the PDSN, the PPP framing, byte stuffing, PPP de-framing, and byte de-stuffing are performed by hardware. The other functions of the PDSN for packet processing are performed as per the method utilizing software in packet processing, as described above with reference to FIGs. 1 and 2. Therefore, in the following description, a detailed description of a process according to the method described above with reference to FIGs. 1 and 2 in cooperation with a processing according to the present invention will be omitted. The PDSN [[109]] 210 shown in FIG. 5 is is not identical to the PDSN 109a shown in FIG. 2 for several reasons, including the fact that the PDSN 109a shown in FIG. 2 does not include generators 230 and 240. However, there are some similarities, for example the PDSN [[109]] 210

of FIG. 5 and the PDSN 109a of FIG. 2 both include a compression/encryption unit 214.

[0054] FIG. 5 is a block diagram showing a detailed construction of a Packet Data Serving Node (PDSN) of an embodiment of the present invention. As shown in FIG. 5, the PDSN [[109]] 210 includes a PPP frame generator 230, which performs PPP framing and byte stuffing, and an IP frame generator 240, which performs PPP de-framing and byte de-stuffing, in addition to the configuration of the PDSN as described above with reference to FIG. 2. The PDSN [[109]] 210 shown in FIG. 5, in accordance with the principles of the present invention, includes the configuration of the PDSN 109a shown in FIG. 2 and also the the PPP Frame Generator 230 and IP Frame Generator 240.

[0059] Meanwhile, an IP frame from the end host 115 is received by the router 111 through the Internet 113, and the router 111 provides the IP frame to the network controller 212 of the PDSN [[109]] 210. The network controller 212 stores the IP frame from the router 111 in the packet memory 619. The IP frame stored in the packet memory 619 is converted into a PPP frame by the PPP frame generator 230, which is then stored in the packet memory 619. The network controller 212 adds a GRE header and an IP header to the PPP frame stored in the packet memory 619 and then provides the frame with the headers to the MAC 211. The MAC 211 adds an MAC header to the frame and then transmits it to the BSC/PCF 105.

[0060] As described above, the network controller 212 utilizes software in converting the frame provided by the MAC 211 into the PPP frame and storing the PPP frame. Also, the network

controller 212 utilizes software in converting the PPP frame generated by ~~means of~~ the IP frame generator 240 ~~from the MAC 211~~ into a frame which can be provided to the BSC/PCF 105.

[0061] A receiving (Rx) ring descriptor 611 and a transmitting (Tx) ring descriptor 612 are included in the packet memory 619. The packet memory 619 includes Rx and Tx ring descriptors 611 and 612 for stuffing, Rx and Tx ring descriptors 613 and 614 for de-stuffing, Rx and Tx PPP stuffing queues 615 and 616, and Rx and Tx PPP de-stuffing queues 617 and 618. The Rx and Tx PPP stuffing queues 615 and 616 are queues used in the PPP frame generator 230 to generate a PPP frame from an IP frame, and the Rx and Tx PPP de-stuffing queues 617 and 618 are queues used in the IP frame generator 240 to generate an IP frame from a PPP frame. The Rx and Tx PPP stuffing queues 615 and 616 and the Rx and Tx PPP de-stuffing queues 617 and 618 are shared by the network controller 212 in its process by software. That is, the network controller 212 stores a PPP frame, which is obtained by eliminating the GRE header and the IP header from the frame received from the MAC 211, in the Tx PPP de-stuffing queue 618, thereby enabling the IP frame generator 240 to read the PPP frame. Meanwhile, the IP frame generated by the IP frame generator 240 is stored in the Rx PPP de-stuffing queue 617, so that the network controller 212 can read the IP frame and transmits it to the router 111. Meanwhile, the IP frame provided to the network controller 212 from the router 111 is stored in the Tx PPP stuffing queue 616, so that the PPP frame generator 230 can read the IP frame. The PPP frame generated by the PPP frame generator 230 is stored in the [[Tx]] Rx PPP stuffing queue [[616]] 615, so that the network controller 212 can read the PPP frame and transmits the PPP frame to the BSC through the MAC 211. The Rx and Tx ring descriptors 611

and 612 reduce software load on the framing and stuffing of the PPP frame generator 230 for generating the PPP frame. The Rx and Tx ring descriptors 613 and 614 reduce software load on the de-framing and de-stuffing of the IP frame generator 240 for generating the IP frame.

[0063] The first PCI interface 631 has both master and slave functions and interfaces data received or transmitted through the bus. In this case, the bus may be a PCI bus which employs data transmission of 32 bits and 66 megahertz (MHz). The first DMA controller 632 performs data communication with the above-described configuration through the first PCI interface 631. That is, the first DMA controller 632 receives an IP frame, which ~~has performed framing and stuffing~~ is to be framed and stuffed, through the first PCI interface 631 from the Tx PPP stuffing queue 616, and stores the IP frame in the first reception buffer 633. Further, the first DMA controller 632 reads a PPP frame, which has been framed and stuffed, from the first transmission buffer 634, and stores the PPP frame in the Rx PPP stuffing queue 615 through the first PCI interface 631. The first reception buffer 633 temporarily stores data provided from the first DMA controller 632 for framing and stuffing. The first reception buffer 633 may be a data width conversion First-in First-out (FIFO) buffer which has an input of 32 bits and an output of 8 bits. The byte stuffing processor 635 receives the IP frame stored in the first reception buffer 633 and converts data of the IP frame into a PPP frame. When stuffing of the converted PPP frame is required, the byte stuffing processor 635 performs stuffing of data recorded in an information field of the PPP frame and then stores the stuffed data in the first transmission buffer 634. In contrast, when stuffing of the converted PPP frame is not required, the byte stuffing processor 635 stores the PPP frame intact in the first

transmission buffer 634.

[0066] The second PCI interface 641 has both master and slave functions and interfaces data received or transmitted through the bus. In this case, the bus may be a PCI bus which employs data transmission of 32 bits and 66 MHz. The second DMA controller 642 performs data communication with the above-described configuration through the second PCI interface 641. That is, the second DMA controller 642 receives a PPP frame, which ~~has been~~ is to be de-framed and de-stuffed, through the second PCI interface 641 from the Tx PPP de-stuffing queue 618, and stores the PPP frame in the second reception buffer 644. Further, the second DMA controller 642 reads an IP frame, which has been de-framed and de-stuffed, from the second transmission buffer 643, and stores the IP frame in the Rx PPP de-stuffing queue 617 through the second PCI interface 641. The second reception buffer 644 temporarily stores data provided from the second DMA controller 642 for de-framing and de-stuffing. The second reception buffer 644 may be a data width conversion FIFO buffer which has an input of 32 bits and an output of 8 bits. The byte de-stuffing processor 645 receives the PPP frame stored in the second reception buffer 644 and converts data of the PPP frame into an IP frame. Before generating the IP frame, when de-stuffing of the PPP frame is required, the byte de-stuffing processor 645 performs de-stuffing of the stuffed data recorded in an information field of the PPP frame, thereby restoring the stuffed data to original data. Then, the byte de-stuffing processor 645 inserts the restored data into the original PPP frame, thereby generating an IP frame.

[0116] In a state wherein transmission of packet data is made possible by the predetermined

initialization described above, an IP ~~frame~~ packet data from the end host 115 is provided to the router 111 through the Internet 113 by the request from the MS 101. The router 111 transfers the IP ~~frame~~ packet data to the PDSN 109. The PDSN 109 performs framing and stuffing corresponding to the IP ~~frame~~ packet data, thereby generating a PPP ~~frame~~ packet data required by the MS 101, and then transfers the generated PPP ~~frame~~ packet data to the BSC/PCF 105.

[0121] When the network controller 212 stores [[PPP]] IP packet data to be framed and stuffed together with control data in the Tx PPP stuffing queue 616, the first DMA controller 632 brings the [[PPP]] IP packet data with the control data and stores them in the first reception buffer 633. The PPP control data for the framing and stuffing include information about the size of the PPP frame data, which enables a start flag and an end flag to be easily inserted in the PPP control data. Meanwhile, the first reception buffer 633 must be controlled to output the stored PPP data after recognizing an enable state of each configurative element.